Tensor Network Complexity of Multilinear Maps

Per Austrin
School of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden
austrin@kth.se

Petteri Kaski
Department of Computer Science, Aalto University, Helsinki, Finland
petteri.kaski@aalto.fi

Kaie Kubjas
Department of Mathematics and Systems Analysis, Aalto University, Helsinki, Finland, and Laboratoire d'Informatique de Paris 6, Sorbonne Université, Paris, France
kaie.kubjas@aalto.fi

Abstract

We study *tensor networks* as a model of arithmetic computation for evaluating multilinear maps. These capture any algorithm based on low border rank tensor decompositions, such as $O(n^{\omega+\epsilon})$ time matrix multiplication, and in addition many other algorithms such as $O(n \log n)$ time discrete Fourier transform and $O^*(2^n)$ time for computing the permanent of a matrix. However tensor networks sometimes yield faster algorithms than those that follow from low-rank decompositions. For instance the fastest known $O(n^{(\omega+\epsilon)t})$ time algorithms for counting $3t$-cliques can be implemented with tensor networks, even though the underlying tensor has border rank n^{3t} for all $t \geq 2$. For counting homomorphisms of a general pattern graph P into a host graph on n vertices we obtain an upper bound of $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ where $\mathrm{bw}(P)$ is the branchwidth of P. This essentially matches the bound for counting cliques, and yields small improvements over previous algorithms for many choices of P.

While powerful, the model still has limitations, and we are able to show a number of unconditional lower bounds for various multilinear maps, including:

(a) an $\Omega(n^{\mathrm{bw}(P)})$ time lower bound for counting homomorphisms from P to an n-vertex graph, matching the upper bound if $\omega = 2$. In particular for P a v-clique this yields an $\Omega(n^{\lceil 2v/3 \rceil})$ time lower bound for counting v-cliques, and for P a k-uniform v-hyperclique we obtain an $\Omega(n^v)$ time lower bound for $k \geq 3$, ruling out tensor networks as an approach to obtaining non-trivial algorithms for hyperclique counting and the Max-3-CSP problem.

(b) an $\Omega(2^{0.918n})$ time lower bound for the permanent of an $n \times n$ matrix.

2012 ACM Subject Classification Theory of computation → Models of computation, Theory of computation → Computational complexity and cryptography, Theory of computation → Design and analysis of algorithms

Keywords and phrases arithmetic complexity, lower bound, multilinear map, tensor network

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.7

Related Version A full version of this paper appears at `https://arxiv.org/abs/1712.09630`.

Funding Per Austrin was funded by the Swedish Research Council, Grant 621-2012-4546. Petteri Kaski has received funding from the European Research Council, Grant 338077. Kaie Kubjas was supported by Marie Skłodowska-Curie Grant 748354.

Acknowledgements We are grateful to Andreas Björklund for highlighting branchwidth to us as a natural parameter to generalize from clique-counting to counting homomorphisms.

1 Introduction

One of the cornerstones of the theory of computation is the study of efficient algorithms:

For a function f, how much time is required to evaluate f on given inputs?

Answering this question for almost any specific f is well beyond reach of contemporary tools. For example, it is theoretically possible that canonical NP-complete problems, such as the Circuit Satisfiability problem, can be solved in linear time whereas they are widely believed to require super-polynomial (or somewhat less widely, exponential) time [34, 35, 36]. The main reason for this barrier to quantitative understanding is that it is very hard to prove lower bounds for explicit functions in general models of computation such as circuits or Turing machines. This situation withstanding, a more modest program to advance our understanding of computation is to study restricted models that for many f are simultaneously

(i) general enough to capture the fastest-known algorithms for f, and

(ii) restricted enough to admit proofs of strong *unconditional* time lower bounds for f.

There is a substantial body of work that fits under this program, ranging from the study of low-depth or otherwise restricted circuits (see e.g. [7], Ch. 14) to models of algorithm-design principles such as greedy algorithms, backtracking, or dynamic programming [3, 27], to linear or semidefinite programming relaxations for hard optimization problems [51].

Multilinear maps. One class of functions f that are of substantial importance is the family of *ℓ-linear maps* (*multilinear maps*) from ℓ input vector spaces to an output vector space.[1] Examples range from maps of known near-linear-time complexity in the input size, such as the discrete Fourier transform [24, 72], to maps without known polynomial-time-complexity algorithms, such as the permanent of a matrix [64, 70]. Beyond motivation in numerical multilinear algebra and its applications, recent advances in the study of fine-grained algorithm design and complexity have highlighted the fundamental role of algebraic methods in the fastest-known algorithm designs across a broad range of tasks from graph problems, such as all-pairs shortest paths and k-clique, to parsing and constraint satisfaction problems, such as maximum satisfiability and graph coloring [2, 11, 13, 30, 37, 54, 75, 76].

In this paper, we study the *arithmetic complexity* of evaluating a multilinear map, that is, the number of operations (scalar additions, subtractions, and multiplications) needed to evaluate the map. To set up a baseline, a generic ℓ-linear map from ℓ vector spaces of dimension n to a scalar requires $\Omega(n^\ell)$ scalars to represent the map directly using combinations of basis vectors. Given this complexity of a direct explicit representation, it is a fundamental problem to seek less costly representations, along with associated efficient algorithms that work on the chosen representation.

We propose the systematic study of *tensor networks* on hypergraphs as a framework for fast evaluation of multilinear maps, and show a number of upper and lower bounds on the computational power of tensor networks in the spirit of (i) and (ii) above.

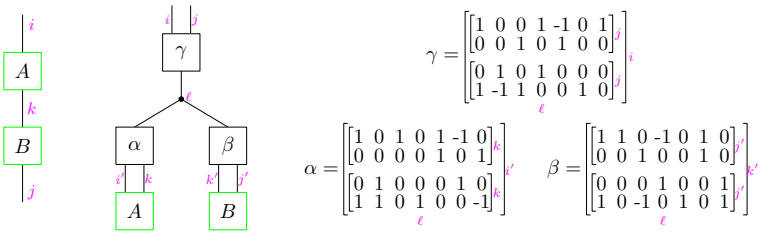
Tensor networks. Tensor networks have a long and rich history which can be traced as far back as 19^{th}-century studies in invariant theory due to Cayley [20, 21], Clebsch [22], Clifford [23], Sylvester [69], and Kempe [40, 41]. Tensor networks are extensively deployed in applications from pure mathematics and theoretical physics [39, 47, 48, 49, 57, 58, 61, 65] to computational physics and chemistry [56, 59, 67]. In theoretical computer science, they appear in various guises including, for example, in the Holant framework [71, 18, 17], in the study of

[1] Multilinear maps with $\ell = 1$ are called *linear*, $\ell = 2$ *bilinear*, $\ell = 3$ *trilinear*, and so forth.

probabilistic graphical models [45, 62], in the study of parallel programming [66], in the study of quantum computing [6], and in the study of arithmetic complexity [8, 19, 26]. Tensor contraction is also emerging as a basic computational primitive in computer hardware [31, 53]. (We refer to the full version of this paper a more detailed discussion.) As the precise definitions are somewhat technical, let us start with a few simple motivating examples and then state our results, with the understanding that precise definitions appear in Section 3.

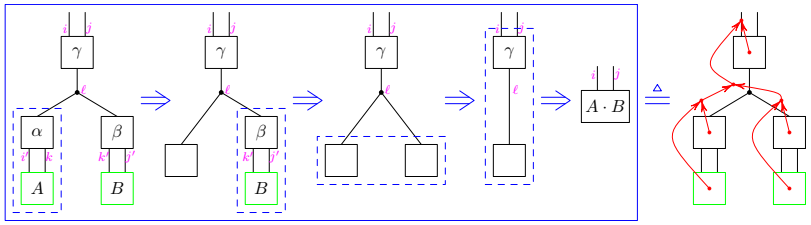
In our setting, a tensor network is a hypergraph in which the vertices are tensors and the hyperedges are called *modes*. Each mode that is incident to a tensor defines a "dimension" for indexing the entries of the tensor – for example, a matrix is a tensor that is incident to two modes, one mode for the rows of the matrix, and the other mode for the columns of the matrix. A network may be simplified by a sequence of *contractions*, where each contraction takes a subset of tensors and replaces it with a single tensor whose entries are obtained as generalized inner products of the entries of the tensors being contracted.

As a first example of these concepts, let us consider the task of multiplying two matrices, A and B. More specifically, let A be a matrix with rows indexed by mode i and columns indexed by mode k, and let B be a matrix with rows indexed by mode k and columns indexed by mode j. We may represent the multiplication task as the tensor network depicted on the left in (1). The result of contracting A and B is a new matrix with rows indexed by i and columns indexed by j, where the entry at each position (i, j) is $\sum_k A_{ik}B_{kj}$. If the three index sets all have size n, then computing $A \cdot B$ by contracting them in such a direct manner uses $\Theta(n^3)$ operations. To obtain faster matrix multiplication, we can rewrite the bare-bones network on the left in (1) using a low-rank decomposition of the matrix multiplication tensor. For example, Strassen's decomposition [68] of 2×2 matrix multiplication can be represented using the second network in (1). Note that the index i used by A and the result has been separated into two distinct indices i and i', and similarly for j and k.



(1)

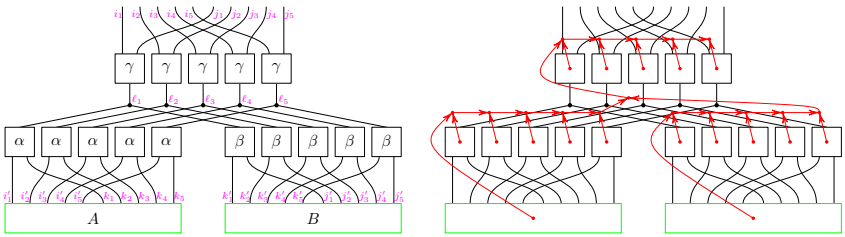
We can *execute* the network by succesively contracting groups of vertices. In (2) we see the process of successively contracting pairs of tensors in a carefully chosen order, until only a single tensor – the result of the computation – remains. Such an execution can be naturally represented by a rooted binary tree, as shown on the right in (2), where the tensors of the network form the leaves, and each internal node represents the result of contracting its two children. To summarize, a tensor-network algorithm is specified by providing (a) a tensor network that when contracted yields the desired result, *and* (b) an execution tree indicating the order in which to contract tensors in the network.



(2)

The *cost* of performing one of the contractions in an execution is the product of the lengths of the modes used by any tensor involved in the contraction. This simply measures (up to a constant factor) the number of arithmetic operations (additions/multiplications) used to compute the result by a direct, naïve computation that does not depend on the values of the tensors. For example, the contraction of α and A in the first step of (2) has cost 28 because it involves the three modes i' (length 2), k (length 2) and ℓ (length 7).

We observe that cost is data-oblivious – the tensor α is fixed with many zero-entries but these entries still contribute to the cost. Indeed, in many cases there may be faster ways of evaluating a contraction than to evaluate it naively, and just like we saw above, this can often be dealt with by rewriting the network appropriately. For instance, consider now the multiplication of two $2^k \times 2^k$ matrices. Because the family of matrix multiplication tensors is closed under Kronecker products, this operation may be computed by a tensor network like the one shown in (3) (depicting the case $k = 5$), where α, β and γ are as in (2). The rows/columns of the matrices are now indexed by k-tuples of bits. The execution of this network contracts one $\alpha/\beta/\gamma$ tensor at a time, which lets us keep the cost low. For example, the first contraction of A with the first α block has a cost of $2^k \cdot 2^k \cdot 7$, and results in a tensor of size $2^{k-1} \times 2^{k-1} \times 7$, then the next contraction has a cost of $2^{k-1} \cdot 2^{k-1} \cdot 7^2$ and produces a result of size $2^{k-2} \times 2^{k-2} \times 7 \times 7$, and so on, until the contraction with the last α block which has a cost of $2 \cdot 2 \cdot 7^k = O(7^k)$, and all the contractions in the execution have cost bounded by this, meaning that we get a total running time of $O(k7^k) = O(N^{\log_2 7} \log N)$ for $N \times N$ matrices.[2]



(3)

This type of argument can capture any algorithm based on a low-rank decomposition of the underlying tensor of the multilinear map, and indeed, this enables $O(n^\omega)$-time[3] matrix multiplication using tensor networks. Beyond simple low-rank decompositions, which always give rise to "star-like" networks as in (3), there are many interesting algorithms that can be captured using networks with a more complicated topology. For instance, many maps of substantial importance have a layered structure that decomposes the map to a sequence of elementary maps. A canonical example is the discrete Fourier transform (DFT), which for a smooth composite order such as 2^k, can be decomposed into a fast Fourier transform (FFT) that consists of a sequence of k transforms of order 2 interleaved with diagonal-matrix multiplications of twiddle factors [24, 72].

---

[2] In fact, a more careful analysis gives running time $O(N^{\log_2 7})$.

[3] Throughout the paper, $\omega = \omega(\mathbb{F})$ denotes the infimum over all e such that the arithmetic complexity of multiplying two $n \times n$ matrices is $O(n^e)$. While the value of ω may depend on the underlying field $\mathbb{F}$, we tacitly ignore this, since the field is fixed throughout the paper. For all fields it is known that $2 \leq \omega < 2.3728639$ [50, 73].

1.1 Our results

Starting with motivation (i) and seeking to express existing fastest-known algorithms as executions of tensor networks by a sequence of contractions, we show upper bounds for a number of natural problems. Beyond standard linear settings such as the FFT, not only do tensor networks realize classical bilinear settings such as Abelian group algebra products and fast matrix multiplication algorithms based on low tensor rank, they are powerful enough to capture a substantial number of higher-linearity applications, including Ryser's algorithm for matrix permanent [64], and the *Kruskal operator* [43, 46], which underlies realization of rank-decompositions for tensor rank [44] and current fastest algorithms for detecting outlier correlations [38].

One problem for which tensor networks turn out to be particularly useful is counting homomorphisms from a fixed pattern graph P to a large host graph G on n vertices. The most well-studied such problem is when P is a k-clique. For this problem, the currently fastest algorithm runs in time roughly $O(n^{\omega k/3})$ (with variations in the exponent depending on $k \bmod 3$) [54, 30]. For general P, it is known that the problem can be solved in $O(n^{\mathrm{tw}(P)+1})$ time [28], where $\mathrm{tw}(P)$ is the treewidth of P. We show that tensor networks can solve the problem in $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ time, where $\mathrm{bw}(P)$ is the *branchwidth* of P. For P a k-clique we have $\mathrm{bw}(P) = \lceil 2k/3 \rceil$ so this almost recovers the $O(n^{\omega k/3})$ running time, and in this case we can slightly improve the running time to recover the $O(n^{\omega \lfloor k/3 \rfloor + (k \bmod 3)})$ time of Nešetřil and Poljak [54]. In the case of general P, this improves on the treewidth-based bound for graphs with $\mathrm{bw}(P) \leq 2(\mathrm{tw}(P)+1)/\omega$ (and in particular if $\omega = 2$ it is always as fast as the treewidth-based bound, ignoring the ϵ). By recent results of Curticapean, Dell, and Marx [25], fast algorithms for homomorphism-counting can be used to obtain fast algorithms for counting subgraphs of G isomorphic to P, and in some cases our new branchwidth-based bound leads to an improvement; for example, for counting paths of lengths of length 7, 8 or 9, we get a running time of $O(n^{3\omega/2+\epsilon}) < O(n^{3.56})$ compared to $O(n^4)$ using the treewidth-based bound, whereas for very long paths it is not clear whether we would need $\omega = 2$ in order for these bound to improve on the treewidth-based bound. Previous work that combines branch decompositions and fast matrix multiplication includes Dorn [29] and Bodlaender *et al.* [15].

Further applications captured by tensor networks are the set covering and set partitioning frameworks via fast zeta and Möbius transforms that underlie the current fastest algorithms for graph coloring [13] and its generalizations such as computing the Tutte polynomial [10, 11]. To summarize, we have the following compendium theorem of upper bound results.

▶ **Theorem 1.1.** *We have the following upper bounds on arithmetic complexity via tensor networks:*

1. $O(n^{\omega+\epsilon})$ *for the matrix multiplication map of two* $n \times n$ *matrices.*
2. $O(n^{(\omega+\epsilon)\lfloor v/3 \rfloor + (v \bmod 3)})$ *for counting* v*-cliques in an* n*-vertex graph.*
3. $O(n^{(\omega+\epsilon)\,\mathrm{bw}(P)/2})$ *for counting homomorphisms of a fixed pattern (hyper)graph* P *into a (hyper)graph on* n *vertices.*
4. $O(\max(n^{\lceil \ell/2 \rceil(\omega+\epsilon-1)}r,\, n^{2\lceil \ell/2 \rceil} r^{\omega+\epsilon-2}))$ *for the Kruskal operator of* ℓ *matrices of shape* $n \times r$*.*
5. $O(2^k k)$ *for the discrete Fourier transforms for the Abelian groups* $\mathbb{Z}_{2^k}$ *and* $\mathbb{Z}_2^k$*.*
6. $O(2^k k)$ *for group algebra products on* $\mathbb{F}[\mathbb{Z}_{2^k}]$ *and* $\mathbb{F}[\mathbb{Z}_2^k]$ *when* 2 *is unit in* $\mathbb{F}$*.*
7. $O(2^k k)$ *for the semigroup algebra product on* $\mathbb{F}[(\{0,1\}^k, \subseteq, \cap, \cup)]$*.*
8. $O(2^n n)$ *for the permanent of an* $n \times n$ *matrix.*

Above $\epsilon > 0$ *is an arbitrary constant.*

Perhaps the most interesting application above is the v-clique problem, which suggests that one should seek to pursue generalizations to v-vertex hypercliques of $\binom{v}{k}$ hyperedges with $v > k \geq 3$. Indeed, subgraph counting is a problem that has received substantial interest over the years (e.g. [37, 54, 5, 4, 30, 12, 14, 77, 74, 33, 32, 55, 42, 25]), but progress in the particular case of v-clique has been stuck to the extent that the problem has attracted notoriety as a hardness assumption in fine-grained complexity [1, 2]. Beyond the study of cliques, hypercliques, and subgraph counting, nontrivial algorithms for such forms would have immediate applicability, for example, in the study of maximum constraint satisfaction problems (Max-CSP) for constraints of width $k \geq 3$; cf. Williams [75] for the case $k = 2$. One of the main goals of our subsequent lower bounds is to rule out tensor networks as a candidate to yield improved algorithms in this setting.

Turning from upper bounds to lower bounds and motivation (ii), tensor networks are restricted enough to enable nontrivial lower bounds for many multilinear maps. To begin with, an immediate limitation of tensor networks is that all the intermediate results during the execution of a network are multilinear, and the execution of a network can be simulated by a multilinear circuit. Raz [60] shows that multilinear formulas cannot compute the determinant of an $n \times n$ matrix in a polynomial number of operations in n, even though polynomial-size general circuits are known for the determinant (cf. [9, 16, 63]).

It turns out that considerably stronger lower bounds can be shown for tensor networks. In particular, we establish essentially tight lower bounds (subject to the assumption $\omega = 2$) for arithmetic complexity via tensor networks of P-homomorphism counting and the Kruskal operator. Furthermore, we rule out the possibility of any nontrivial algorithm designs via tensor networks for counting cliques in hypergraphs. The following theorem collects our main lower-bound results, and should be contrasted with the upper bounds in Theorem 1.1.

▶ **Theorem 1.2.** *We have the following lower bounds on arithmetic complexity via tensor networks:*

1. $\Omega(n^{\mathrm{bw}(P)})$ *for the multilinear form corresponding to P-homomorphism counting. In particular, this yields a lower bound of $\Omega(n^{\lceil 2v/3 \rceil})$ for counting cliques of size v, and a lower bound of $\Omega(n^v)$ for counting hypercliques of size v.*
2. $\Omega(\max(n^\ell, n^{\lceil l/2 \rceil} r))$ *for the Kruskal operator of ℓ matrices of shape $n \times r$.*
3. $\Omega(\binom{n}{n/3})$ *for the determinant or permanent of an $n \times n$ matrix.*

We remark that [52] independently showed that the border rank of the v-hyperclique tensor is $\Omega(n^v)$; our $\Omega(n^v)$ lower bound for tensor networks strengthens that. One may wonder about the gap between the bounds of $\Omega(\binom{n}{n/3})$ and $O(2^n n)$ for the permanent. As we explain below, our lower bound methods are inherently rank-based and cannot go beyond $\binom{n}{n/3}$. A curious point is that it is not immediately clear whether tensor networks can even achieve $O^*(2^n)$ time for the determinant, and we do not know whether or not this is possible.

1.2 Overview of proof ideas

As a running example in this overview, we consider the 6-linear forms $A : \mathbb{F}^{[n]\times[n]} \times \mathbb{F}^{[n]\times[n]} \times \ldots \times \mathbb{F}^{[n]\times[n]} \to \mathbb{F}$ taking as input 6 matrices of size $n \times n$, defined by

$$A(X^{(1)}, X^{(2)}, X^{(3)}, X^{(4)}, X^{(5)}, X^{(6)}) = \sum_{i,j,k,\ell \in [n]} X^{(1)}_{ij} X^{(2)}_{ik} X^{(3)}_{i\ell} X^{(4)}_{jk} X^{(5)}_{j\ell} X^{(6)}_{k\ell} \,. \tag{4}$$

If χ is the adjacency matrix of a loopless graph G, then $A(\chi, \chi, \chi, \chi, \chi, \chi)$ counts the number of 4-cliques in the graph. Associated with A is the 6-tensor $T(A)$ of size $n^2 \times n^2 \times \cdots \times n^2$,

where each of the 6 modes is indexed by a pair $(i, j) \in [n] \times [n]$, and the value at a given position is the coefficient of the corresponding term in A. Concretely,

$$T(A)_{i_1 j_1, i_2 k_2, i_3 \ell_3, j_4 k_4, j_5 \ell_5, k_6 \ell_6} = \begin{cases} 1 & \text{if } i_1 = i_2 = i_3,\ j_1 = j_4 = j_5,\ k_2 = k_4 = k_6 \wedge \ell_3 = \ell_5 = \ell_6, \\ 0 & \text{otherwise.} \end{cases}$$

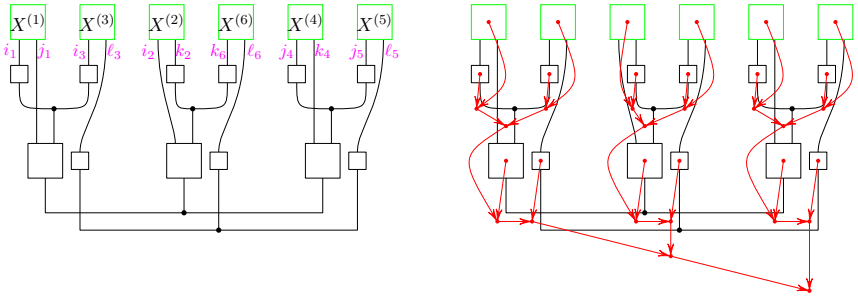
Upper bounds. Most, but not all, of the families of multilinear maps we consider are closed under taking Kronecker products. For instance, consider the 4-clique counting form (4) for an n-vertex graph and its associated tensor $T(A)$. Then for any $k \geq 1$, the tensor associated with the 4-clique counting form in n^k-vertex graphs is $T(A)^{\otimes k}$, the k-fold Kronecker product of $T(A)$ with itself. We write $A^{\otimes k}$ for the associated map. With this in mind, it is natural to seek general constructions that, given an efficient evaluation of some map A, yields an efficient evaluation of $A^{\otimes k}$.

We give such a construction, and show that the cost of the best tensor network execution for $A^{\otimes k}$ is essentially submultiplicative in a quantity that we call the *amortized cost* of an execution. For tensors of order at most 3, the notion of amortized cost essentially captures the rank of $T(A)$, but for higher-order tensors, the amortized cost may be significantly smaller than the rank. Roughly speaking, the amortized cost of a step in an execution of a map A is: (i) equal to the normal cost if the operation involves the contraction of two tensors that both depend on some input variables of A, but (ii) equal to the size of the result if only one of the tensors involved in the contraction depends on the input variables of A. A precise definition appears in Section 4. Our general upper bound for the cost of $A^{\otimes k}$ can, somewhat informally, be stated as follows.

▶ **Theorem 1.3** (Submultiplicativity of cost, informal statement). *If a multilinear map A has a tensor network execution consisting of s steps, each with cost at most c and amortized cost at most a, then $A^{\otimes k}$ has a tensor network execution consisting of at most $k \cdot s$ steps, each with cost at most $a^{k-1} \cdot c$.*

An immediate corollary of this is that we can capture any algorithm for $A^{\otimes k}$ based on a low-rank decomposition of $T(A)$ (Corollary 4.2). For example, this implies that tensor networks can multiply $n \times n$ matrices in $O(n^{\omega+\epsilon})$ time.

However, returning to our running example form (4), as we explain below the tensor $T(A)$ has rank n^4, meaning that Corollary 4.2 only yields a trivial upper bound. This is where the full generality of Theorem 1.3 comes in. Consider the form (4) for graphs on some constant number n_0 of vertices. As it turns out, we can design a network and an associated execution for this form, depicted in (5) and explained in more detail in the full version of this paper, with an execution of cost n_0^{2e+3} and amortized cost n_0^{e+1}, where n_0^e is the rank of the tensor associated with $n_0 \times n_0$ matrix multiplication. Picking n_0 to be a large enough *constant* so that e is approximately ω, and letting k be such that n is approximately n_0^k, we obtain via Theorem 1.3 an $O(n^{\omega+\epsilon+1})$ time upper bound for (4).



(5)

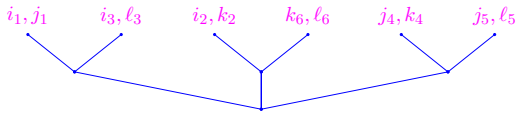
Lower bounds. Just like many other arithmetic complexity lower bounds, our lower bounds boil down to establishing lower bounds on the rank of certain matrices.

To establish a lower bound on the *rank* of $T(A)$, we *flatten* it to a matrix and analyze the rank of that matrix. There are 2^5 possible bipartitions of the set of 6 modes of $T(A)$, and the lower bound on the rank of $T(A)$ that we obtain is the maximum of the ranks of the resulting matrices. Using this method it is easy to establish that for our example form (4), the rank of $T(A) = n^4$. That this is an upper bound follows from (4), and that it is a lower bound follows by considering the bipartition taking variables $X^{(1)}$ and $X^{(6)}$ as row indices, and the other 4 variables as column indices. The resulting $n^4 \times n^8$ matrix has full rank.

Tensor networks are more versatile and can be more efficient than low-rank decompositions of $T(A)$. Nevertheless, we show limitations on this versatility. In particular we show that every tensor network execution for A induces a tree in which the leaves are the inputs of A and all internal vertices have degree 3. We call this a *socket tree*. Each edge in a socket tree induces a bipartition of the variables and our key technical lemma is to show that for each such bipartition, the rank of the corresponding flattening of $T(A)$ is a lower bound on the cost of the execution that gave rise to the tree. Thus, to obtain a lower bound for the cost of a specific execution, we consider the maximum rank obtained over all edges of the corresponding socket tree, and to lower bound the cost of every tensor network execution, we minimize this quantity over all possible socket trees. We refer to the resulting quantity as the *socket width* of A, denoted $w(A)$ (formal definition appears in Section 5). Our general lower bound can thus be phrased as follows, where $c(A)$ denotes the minimum cost of a tensor network for evaluating A (formal definition appears in Section 3).

▶ **Theorem 1.4.** *For every multilinear map A, it holds that $c(A) \geq w(A)$.*

Indeed, for our running example (4), there are low-width socket trees establishing that $w(A) \leq n^3$, see (6). However, that bound is tight: our $\Omega(n^{\lceil 2 \cdot 4/3 \rceil}) = \Omega(n^3)$ lower bound for the $\binom{4}{2}$-linear form (Theorem 1.2) is obtained by proving that $w(A) \geq n^3$ and appealing to Theorem 1.4.



(6)

1.3 Organization of this paper

The present conference abstract contains only the key definitions and technical results underlying our main theorems. All the upper and lower bounds for the arithmetic complexity of specific multilinear maps together with their proofs can be found in the full version of this paper. Section 2 recalls preliminaries on tensors and multilinear maps. In Section 3, tensor networks, execution and cost of a tensor network, and cost of a multilinear map are defined. Section 4 presents our main upper-bound result on submultiplicativity of cost. In Section 5, a general lower bound on the cost of evaluating a multilinear map using tensor network is obtained; this lower bound is expressed in terms of the socket-width of a multilinear map.

2 Preliminaries on tensors and multilinear maps

This section sets up our notation for tensors and multilinear maps. Throughout the paper $[n]$ denotes $\{1, 2, \ldots, n\}$ and $\mathbb{F}$ denotes an arbitrary fixed field. We work with tensors and multilinear maps relative to fixed bases for the respective vector spaces over $\mathbb{F}$.

Modes, indexing, and positions. We will work with the following convention of positioning individual entries inside a tensor. Let E be a finite set of *modes*. Associate with each mode $e \in E$ a finite nonempty *index set* $J(e)$. In this case we say that E is a set of *indexed* modes. The *length* of e is $|J(e)|$. A *position* is an element $j \in \prod_{e\in E} J(e)$. Let us write $J(E) = \prod_{e\in E} J(e)$ for the set of all positions with respect to the indexed modes E. In the special case that the set of modes E is empty we define the set of positions $J(E)$ to consist of a single element.

Tensors, matrices, vectors, and scalars. Let E be a set of indexed modes. A *tensor* with respect to E is a mapping $T : J(E) \to \mathbb{F}$. Equivalently, we write $T \in \mathbb{F}^{J(E)}$ to indicate that T is a tensor with respect to the indexed modes E. We view the set $\mathbb{F}^{J(E)}$ of all tensors over E as a vector space over $\mathbb{F}$ with addition and scalar multiplication of tensors defined entrywise. We say that $|E|$ is the *order* of the tensor. A tensor of order zero is called a *scalar*, a tensor of order one is called a *vector*, and a tensor of order two is called a *matrix*. The *volume* of the tensor is $|J(E)|$. The tuple $(|J(e)| : e \in E)$ is the *shape* of the tensor. It is convenient to use the "$\times$"-symbol to punctuate the shape of a tensor; that is, instead of writing, say $(2, 3, 4)$ for the shape, we write $2 \times 3 \times 4$. For a position $j \in J(E)$ and a tensor $T \in \mathbb{F}^{J(E)}$, we say that $T_j \in \mathbb{F}$ is the *entry* of T at j.

A *flattening* of T induced by a bipartition $E_1 \cup E_2 = E$ of the modes of T is a $|J(E_1)| \times |J(E_2)|$ matrix M where, for $j_1 \in J(E_1)$ and $j_2 \in J(E_2)$ we have $M_{j_1,j_2} = T_{j_1 j_2}$. Given two order ℓ tensors $S \in \mathbb{F}^{[n_1]\times[n_2]\times\cdots\times[n_\ell]}$ and $T \in \mathbb{F}^{[m_1]\times[m_2]\times\cdots\times[m_\ell]}$, their *Kronecker product* $S \otimes T$ is a tensor in $\mathbb{F}^{[n_1 m_1]\times[n_2 m_2]\times\cdots\times[n_\ell m_\ell]}$ defined by

$$(S \otimes T)_{m_1(i_1-1)+j_1, m_2(i_2-1)+j_2, \ldots, m_\ell(i_\ell-1)+j_\ell} = S_{i_1,i_2,\ldots,i_\ell} T_{j_1,j_2,\ldots,j_\ell}.$$

Multilinear maps. Let $E_1, E_2, \ldots, E_\ell, E'$ be pairwise disjoint sets of indexed modes such that $E_1, E_2, \ldots, E_\ell$ are nonempty. A map $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ is an *ℓ-linear map* if A is linear with respect to each of its ℓ inputs individually. In particular, a 1-linear map is a linear map. A multilinear map that gives a scalar output is a multilinear *form*. In particular, A is a form if and only if E' is empty.

The tensors of a multilinear map. For an ℓ-linear map $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$, we define two slightly different tensors $T(A)$ and $\hat{T}(A)$. Both are indexed by $J(E_1 \cup E_2 \cup \ldots \cup E_\ell \cup E')$ and at position $j_1 j_2 \ldots j_\ell j'$ take the value

$$T(A)_{j_1 j_2 \ldots j_\ell j'} = \hat{T}(A)_{j_1 j_2 \ldots j_\ell j'} = A\big(e^{(j_1)}, e^{(j_2)}, \ldots, e^{(j_\ell)}\big)_{j'},$$

where $e^{(j_i)} \in \mathbb{F}^{J(E_i)}$ denotes the tensor with a 1 in position j_i and 0s in all other position. The difference between $T(A)$ and $\hat{T}(A)$ is their shape. The shape of $T(A)$ is $|J(E_1)| \times |J(E_2)| \times \cdots \times |J(E_\ell)| \times |J(E')|$, except if A is a form in which case the $|J(E')|$ part is omitted. Thus $T(A)$ is of order $\ell + 1$ (or ℓ if A is a form). The shape of $\hat{T}(A)$ is $(|J(e)| : e \in E_i, i \in [\ell + 1])$, thus its order is $|E_1| + |E_2| + \cdots + |E_\ell| + |E'|$.

In other words, each mode of $T(A)$ corresponds to one of the inputs of A, or the output. These inputs are in turn sets of indexed modes so may contain more "fine-grained" structure, but this information is lost at the level of granularity of $T(A)$. When working with tensor networks for evaluating A, we need to keep track of the fine-grained mode structure because this is in many cases what allows us to construct efficient algorithms, hence in most parts of the paper we are more interested in the tensor $\hat{T}(A)$ which contains this structure.

On the other hand, $\hat{T}(A)$ does not contain information about which modes are grouped together to form the inputs and output of A, and this information is also important. This leads us to the notion of sockets, defined next.

Sockets. Let us study the tensor $\hat{T}(A)$ with respect to the map A. We say that the modes in $E_1 \cup E_2 \cup \cdots \cup E_\ell$ are the *input* modes of $\hat{T}(A)$, and the modes in E' are the *output* modes of $\hat{T}(A)$ with respect to A. Let us say that $E_1, \ldots, E_\ell$ are the *input sockets* of $\hat{T}(A)$ with respect to A. Similarly, E' is the *output socket* in $\hat{T}(A)$ with respect to A. In particular, the output socket is empty if and only if A is a form. To describe a socketing of the modes of a tensor, it is convenient to use parentheses to group a "$\times$"-punctuated shape of a tensor into sockets, see also Section 2.

Let $\hat{T}$ be a tensor over a set of indexed modes E. Any tuple $\mathcal{E} = (E_1, E_2, \ldots, E_\ell, E')$ of subsets of E that partitions E with $E_1, E_2, \ldots, E_\ell$ nonempty defines an ℓ-linear map $A_{\mathcal{E}}(\hat{T}) : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ with $\hat{T}(A_{\mathcal{E}}(\hat{T})) = \hat{T}$. In this case the tuple $(E_1, E_2, \ldots, E_\ell)$ gives the input sockets of T and E' is the output socket of $\hat{T}$ with respect to $A_{\mathcal{E}}(\hat{T})$. We thus conclude that two multilinear maps A_1, A_2 may have the same base tensor $\hat{T}(A_1) = \hat{T}(A_2)$, and from a tensor $\hat{T}$ one may obtain different multilinear maps by varying how the modes of $\hat{T}$ are assigned to input and output sockets.

The form of a multilinear map. Let A be a multilinear map with a nonempty output socket. We can turn A into a multilinear form $F(A)$ by turning its output socket into an input socket. Let us say that $F(A)$ is the *multilinear form* of A. We also set $F(A) = A$ when A is a multilinear form.

3 Tensor networks

This section defines tensor networks and the cost of a multilinear map.

Networks. A *network* (or *diagram*) consists of a finite set V of *vertices*, a finite set E of *hyperedges*, an *incidence relation* $I \subseteq V \times E$, and a *boundary* $B \subseteq E$. A network is *nondegenerate* if every hyperedge is incident to at least one vertex. In what follows we assume that all networks are nondegenerate. A hyperedge $e \in E$ is a *loop* if $e \notin B$ and e is incident to exactly one vertex.

For a vertex $v \in V$, let us write $I(v) = \{e \in E : (v, e) \in I\}$ for the set of hyperedges incident to v. Dually, for an hyperedge $e \in E$, let us write $I(e) = \{v \in V : (v, e) \in I\}$ for the set of vertices incident to e. For a network D, we write $V(D)$, $E(D)$, $I(D)$, and $B(D)$ to refer to the vertices of D, the hyperedges of D, the incidence relation of D, and the boundary of D, respectively.

Induced networks. For a network D and a nonempty subset $W \subseteq V(D)$, the *induced* network $D[W]$ consists of the vertices in W together with the hyperedges of D that are

incident to at least one vertex in W; the boundary of $D[W]$ consists of all hyperedges that are at the boundary of D or incident to a vertex outside W. Formally,

$$\begin{aligned} V(D[W]) &= W\,, \\ E(D[W]) &= \{e \in E(D) : \exists w \in W \text{ s.t. } (w,e) \in I(D)\}\,, \\ I(D[W]) &= I(D) \cap (V(D[W]) \times E(D[W]))\,, \\ B(D[W]) &= (B(D) \cap E(D[W])) \cup \{e \in E(D[W]) \colon \exists v \in V(D) \setminus W \text{ s.t. } (v,e) \in I(D)\}. \end{aligned} \tag{7}$$

For a vertex $v \in V$, we abbreviate $D[v] = D[\{v\}]$. Note that the boundary of $D[v]$ consists of all non-loop hyperedges incident to v in D.

Tensor networks. Let D be a network. We *index* D by associating with each hyperedge $e \in E$ an *index set* $J(e)$ of size $\ell(e)$. Induced networks inherit indexing by restriction. Next we associate with each vertex $v \in V$ a tensor $T(v) \in \mathbb{F}^{J(I(v))}$. We say that D equipped with the tensors $(T(v))_{v \in V}$ is a *tensor network*.

The *value* of a tensor network D, or the tensor *represented by* D, is a tensor $T(D) \in \mathbb{F}^{J(B)}$, defined for all $i \in J(B)$ by

$$T(D)_i = \sum_{j \in J(E(D) \setminus B)} \prod_{v \in V} T(v)_{ij}\,. \tag{8}$$

Observe that in (8) the positions i and j together identify a unique entry of $T(v)$ by projection to $J(I(v))$. The value of a tensor network with an empty boundary is a scalar.

Contracting tensors. Let D be a tensor network and let $W \subseteq V(D)$ be a nonempty set of vertices. Let w be a new element not in V. We may *contract* W in D to obtain a tensor network D/W by replacing the sub-network $D[W]$ in D with the single vertex w whose associated tensor $T(w)$ is the tensor represented by $D[W]$. Formally,

$$\begin{aligned} V(D/W) &= (V(D) \setminus W) \cup \{w\}\,, \\ E(D/W) &= E(D) \setminus (E(D[W]) \setminus B(D[W]))\,, \\ I(D/W) &= (I(D) \setminus I(D[W])) \cup \{(w,e) : e \in B(D[W])\}\,, \\ B(D/W) &= B(D)\,, \\ T(w) &= T(D[W])\,. \end{aligned} \tag{9}$$

The *cost* of contracting W in D is $c(D,W) = \prod_{e \in E(D[W])} |J(e)|$. The value of a tensor network is invariant under contraction, i.e., for all nonempty $W \subseteq V(D)$ it holds that $T(D) = T(D/W)$.

Execution and cost of a tensor network. To compute the tensor $T(D)$ from a given tensor network D, we may proceed by a sequence of contractions on D. Such a process is called executing D, and the cost of D is the cost of a minimum-cost execution of D.

More precisely, let $D = D_0$ be a tensor network with at least one tensor. For $k = 1, 2, \ldots, t$, select a nonempty subset $W_{k-1} \subseteq V(D_{k-1})$ such that W_{k-1} has at least two tensors or consists of a single tensor with a loop. Set $D_k = D_{k-1}/W_{k-1}$ and observe that the number of tensors and/or modes decreases by at least one in the contraction. Suppose that D_t is loopless and consists of a single tensor. We say that such a sequence of contractions is an *execution* of D in t *steps*. The *cost* of the execution is $\max_{k=1,2,\ldots,t} c(D_{k-1}, W_{k-1})$. The cost of an execution in zero steps is defined to be 0.

It is immediate that D has at least one execution and every execution consists of at most $2|V(D)|-1$ steps. By invariance under contractions, we have $T(D_t) = T(D)$. The *cost* $c(D)$ of D is the cost of a minimum-cost execution of D.

An execution of D of cost $c(D)$ immediately translates into an algorithm that computes $T(D)$ using $O(c(D)|V(D)|)$ arithmetic operations in $\mathbb{F}$, since the contraction step $D_k = D_{k-1}/W_{k-1}$ takes $O(c(D_{k-1}, W_{k-1})) \leq c(D)$ time to evaluate, and there are $O(V(D))$ steps.

▶ **Lemma 3.1.** *Let D be a tensor network. There exists a minimum-cost execution of D such that each contracted set has size at most two. Furthermore, if D is loopless, we can assume that each contracted set has size exactly two.*

In what follows we restrict to consider loopless D only. Thus while a general execution may contract arbitrary vertex sets in D in each step, without loss of generality the minimum-cost execution has the structure of a rooted binary tree, whose leaves are the vertices of the tensor network, and each internal vertex is the tensor obtained by contracting its two children.

Cost of a multilinear map. We now define the cost of a multilinear map via the minimum-cost tensor networks (and socketing) for evaluating the map. That is, the cost of a map is defined in terms of the best tensor network that implements the map. More precisely, let

$$A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$$

be an ℓ-linear map. Consider the tensor $\hat{T}(A)$ of A and the associated input sockets $E_1, E_2, \ldots, E_\ell$ and the output socket E'. Let D^* be an arbitrary tensor network such that $T(D^*) = \hat{T}(A)$ and the boundary satisfies $B(D^*) = E_1 \cup E_2 \cup \cdots \cup E_\ell \cup E'$. Modify the network D^* as follows. For each $k = 1, 2, \ldots, \ell$, introduce a new vertex to D^*, make the new vertex incident to each of the modes in the input socket E_k, and associate the new vertex with a tensor $X^{(k)} \in \mathbb{F}^{J(E_k)}$. Remove the modes $E_1 \cup E_2 \cup \cdots \cup E_\ell$ from the boundary of D^*. Let us denote the resulting network by D and call the introduced ℓ new vertices the *socket vertices* of D. We observe that $B(D) = E'$ and $A(X^{(1)}, X^{(2)}, \ldots, X^{(\ell)}) = T(D)$. Furthermore, the cost $c(D)$ is independent of the value of $X^{(k)} \in \mathbb{F}^{J(E_k)}$ for $k = 1, 2, \ldots, \ell$. We say that D is a *realization* of A if it can be obtained from A by this process, and write $\mathscr{D}(A)$ for the set of all tensor network realizations D of A.

The *cost* of the map A is $c(A) = \min_{D \in \mathscr{D}(A)} c(D)$. This minimum exists since the cost of a tensor network is a nonnegative integer and the family $\mathscr{D}(A)$ is nonempty.

4 An upper bound via submultiplicativity

This section presents our main tool for proving upper bounds on the cost of a multilinear map that admits representation as a Kronecker power, namely submultiplicativity of an amortized notion of cost under Kroneckering.

Kronecker power of a multilinear map. Let $E_1, E_2, \ldots, E_\ell, E'$ be pairwise disjoint sets of indexed modes such that $E_1, E_2, \ldots, E_\ell$ are nonempty. Let

$$A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$$

be an ℓ-linear map. For a positive integer k, we define the ℓ-linear map $A^{\otimes k}$ such that its tensor satisfies $T(A^{\otimes k}) = T(A)^{\otimes k}$. Then

$$A^{\otimes k} : \mathbb{F}^{J(E_1)^k} \times \mathbb{F}^{J(E_2)^k} \times \cdots \times \mathbb{F}^{J(E_\ell)^k} \to \mathbb{F}^{J(E')^k}.$$

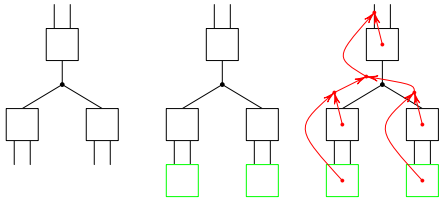
Note that $T(A^{\otimes k}) = T(A)^{\otimes k}$ is the k-fold Kronecker product of $T(A)$ with itself – that is, it has the same order, but the index sets are larger – whereas $\hat{T}(A^{\otimes k})$ is the k-fold outer product of $\hat{T}(A)$ with itself – that is, its index sets have the same sizes, but its order is k times larger.

Amortized cost and submultiplicativity. Let D be a diagram that realizes A and let $\mathcal{T}_D$ be an execution tree for D. For each internal vertex x in $\mathcal{T}_D$ (that is, a vertex obtained by contraction), define the *amortized cost* of x by splitting into the following three cases:

- **(i)** if neither of the two subtrees of x contains a socket vertex, the amortized cost of x is 1;
- **(ii)** if exactly one of the subtrees of x, say, the subtree rooted at y (where x and y are adjacent in $\mathcal{T}_D$), contains at least one socket vertex, the amortized cost of x is the maximum of the volume of the tensor at x and the volume of the tensor at y;[4]
- **(iii)** if both of the subtrees of x contain at least one socket vertex, the amortized cost of x is the cost of the contraction to obtain x.

The *amortized cost* $a(\mathcal{T}_D)$ of $\mathcal{T}_D$ is the maximum of the amortized costs of the internal vertices of $\mathcal{T}_D$. Since the amortized cost of each internal vertex of $\mathcal{T}_D$ is at most its cost, we have $a(\mathcal{T}_D) \leq c(\mathcal{T}_D)$. Furthermore, we observe that the amortized cost of x in case (ii) above may be strictly less than the cost of the contraction to obtain x. In particular, in (ii) the amortized cost is defined *not by the cost of the contraction but rather by volume*. This is because in a k^{th} Kronecker power we can amortize the cost of the aggregate transformation in case (ii) not with a single contraction but with a sequence of k contractions. This observation will form the heart of the proof of Theorem 1.3.

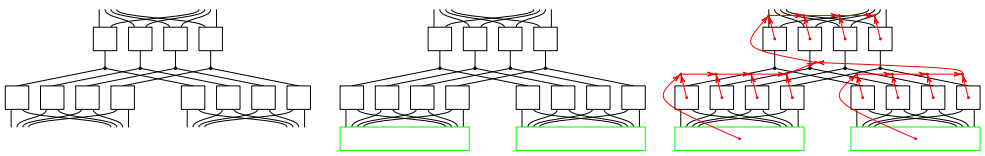
Before proceeding with the proof, let us illustrate the key ideas in visual terms. Let us start with the three illustrations in (10).



(10)

Suppose the leftmost network in (10) is socketed so that the two modes at the top form the output socket, and the four modes at the bottom form two input sockets so that modes in the same socket are incident to the same vertex. In the middle in (10), we have adjoined two socket vertices to the input sockets to obtain a realization D. On the right in (10), we display an execution tree $\mathcal{T}_D$ for D. Observe that the bottom-most internal vertices of $\mathcal{T}_D$, and the top-most internal vertex of $\mathcal{T}_D$, have type (ii). The internal vertex in the center has type (iii). (There are no internal vertices of type (i).) Supposing that all the modes have length at least 2, we also observe that the vertices of type (ii) have amortized cost strictly less than their contraction cost.

Let us now consider the k^{th} power of (10) visually, for $k = 4$:



(11)

[4] Here, it is crucial to note that the volume of the other subtree rooted at x, only containing non-socket vertices, does not contribute directly to the amortized cost of x.

The leftmost network in (11) depicts the k-fold outer product of the network on the left in (10) with itself. Observe that we simply take k copies of the network, but that for the purposes of the visualization we have taken care to draw the k copies of each mode together for the socketing. In the middle in (11), we have adjoined two socket vertices to the input sockets to obtain a realization $D^{\otimes k}$ of $A^{\otimes k}$. On the right in (11), we display an execution tree $\mathcal{T}_{D^{\otimes k}}$ for $D^{\otimes k}$. Observe how each of the internal vertices of type (ii) in $\mathcal{T}_D$ gets expanded to a sequence of k internal vertices in $\mathcal{T}_{D^{\otimes k}}$. This transformation from $\mathcal{T}_D$ to $\mathcal{T}_{D^{\otimes k}}$ is the gist of the following theorem.

▶ **Theorem 4.1** (Formal statement of Theorem 1.3)**.** *Let D be an arbitrary realization of A and let $\mathcal{T}_D$ be an arbitrary execution tree for D. For all positive integers k, we have*

$$c(A^{\otimes k}) \leq a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)\,. \tag{12}$$

Furthermore, this realization of $A^{\otimes k}$ consists of at most $k \cdot |V(D)|$ vertices.

Proof. Let D^* be the subnetwork of D with $T(D^*) = \hat{T}(A)$. That is, D^* is the network induced by all the non-socket vertices of D. Taking k disjoint copies of D^*, we obtain a network whose tensor is $\hat{T}(A^{\otimes k})$. Attaching the resulting network to tensors at sockets gives a realization of $A^{\otimes k}$. Let us write $D^{\otimes k}$ for this realization.

To establish (12), it suffices to construct an execution tree $\mathcal{T}_{D^{\otimes k}}$ for $D^{\otimes k}$ whose cost satisfies $c(\mathcal{T}_{D^{\otimes k}}) \leq a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)$. We construct $\mathcal{T}_{D^{\otimes k}}$ by rewriting $\mathcal{T}_D$ from leaves towards the root to consider the k copies of each vertex in D^*. We start with leaf vertices which are the vertices of $D^{\otimes k}$. We split the process into cases (i), (ii), and (iii) as in the definition of amortized cost. Let x be the internal vertex of $\mathcal{T}_D$ that we are currently considering.

In case (i), we perform the contraction indicated by x in each of the k copies of D^* in $D^{\otimes k}$ individually. This creates k new internal vertices in $\mathcal{T}_{D^{\otimes k}}$ that are all copies of x. We set these k vertices as the vertices that correspond to x in the subsequent steps. Each of these contractions in $\mathcal{T}_{D^{\otimes k}}$ has the same cost as the contraction indicated by x in $\mathcal{T}_D$. This cost is less or equal than $c(\mathcal{T}_D)$.

In case (ii), let y be the child of x in $\mathcal{T}_D$ such that the subtree rooted at y contains a socket vertex, and let z be the other child of x in $\mathcal{T}_D$. There is a single vertex in $\mathcal{T}_{D^{\otimes k}}$ corresponding to y and k identical vertices in $\mathcal{T}_{D^{\otimes k}}$ corresponding to z. We contract these k vertices individually each with the vertex that corresponds to y. This creates k new internal vertices in $\mathcal{T}_{D^{\otimes k}}$, where we set the topmost vertex as the vertex that corresponds to x in the subsequent steps. After the ith step, the corresponding tensor has i copies of modes of x and $k-i$ copies of modes of y. The cost of the contraction in the ith step is the cost of contracting y and z in $\mathcal{T}_D$ multiplied by the the volume of y to the power $k-i$ and the volume of x to the power $i-1$. Since the volumes of x and y are less than or equal to $a(\mathcal{T}_D)$, this cost is less than or equal to $a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)$.

In case (iii), let y and z be the two child vertices of x in $\mathcal{T}_D$. By the structure of the earlier steps, we have that a single vertex in $D^{\otimes k}$ corresponds to y, and similarly for z. We contract these two vertices. This creates one new internal vertex in $\mathcal{T}_{D^{\otimes k}}$, which we set as the vertex that corresponds to x in the subsequent steps. This tensor has k copies of modes of x. The cost of this contraction in $\mathcal{T}_{D^{\otimes k}}$ is the cost of the corresponding contraction in $\mathcal{T}_D$ to the k^{th} power, because both tensors have k copies of all modes compared to y and z. By definition, in case (iii) the amortized cost of contracting y and z is the same as the cost of contracting y and z. Hence the cost of this contraction in $\mathcal{T}_{D^{\otimes k}}$ is less or equal than $a(\mathcal{T}_D)^k \leq a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)$. This rewriting process produces an execution tree $\mathcal{T}_{D^{\otimes k}}$ for $D^{\otimes k}$ with $c(\mathcal{T}_{D^{\otimes k}}) \leq a(\mathcal{T}_D)^{k-1} c(\mathcal{T}_D)$. ◀

An immediate corollary is that tensor networks can use low rank decompositions of $T(A)$ to efficiently evaluate $A^{\otimes k}$.

▶ **Corollary 4.2** (Submultiplicativity of low-rank executions). *Let* $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ *be a multilinear map. Define* $m = \max\{|J(E_1)|, |J(E_2)|, \ldots, |J(E_\ell)|, |J(E')|\}$ *and* $r = \operatorname{rk} T(A)$. *Then* $c(A^{\otimes k}) \leq \max(r, m)^k \min(r, m)$

Proof. By taking a star-like network topology (as in (10)) we get an execution with $a(\mathcal{T}_D) = \max(r, m)$ and cost $c(\mathcal{T}_D) = m \cdot r$. ◀

5 A lower bound for the cost of a multilinear map

In this section, we prove a general lower bound on the cost of evaluating a multilinear map using tensor networks, as defined in Section 3. The lower bound is expressed in terms of the *socket-width* of a multilinear map, which we now proceed to define.

Let $A : \mathbb{F}^{J(E_1)} \times \mathbb{F}^{J(E_2)} \times \cdots \times \mathbb{F}^{J(E_\ell)} \to \mathbb{F}^{J(E')}$ be an ℓ-linear map. A *socket-tree* of A is a tree $\mathcal{T}_S$ whose $\ell + 1$ leaf vertices are the sockets $E_1, E_2, \ldots, E_\ell, E'$ of A and whose internal vertices all have degree exactly 3. Associate with each edge $e = \{x_R, x_C\}$ of $\mathcal{T}_S$ the two subtrees $\mathcal{T}_S(x_R, e)$ and $\mathcal{T}_S(x_C, e)$ obtained by removing e, where $\mathcal{T}_S(x_R, e)$ is the subtree containing x_R and $\mathcal{T}_S(x_C, e)$ is the subtree containing x_C. Let $L(x_R, e)$ be the set of leaves in $\mathcal{T}_S(x_R, e)$ and let $L(x_C, e)$ be the set of leaves in $\mathcal{T}_S(x_C, e)$.

The sets $L(x_R, e)$ and $L(x_C, e)$ are both nonempty and together partition the set of sockets. Consider the flattening $M(\mathcal{T}_S, e)$ of the tensor $T(A)$ such that the modes in $L(x_R, e)$ index the rows and the modes in $L(x_C, e)$ index the columns of $M(\mathcal{T}_S, e)$. The *width* of $\mathcal{T}_S$ at e is the rank of $M(\mathcal{T}_S, e)$, and the *width* of $\mathcal{T}_S$ is $w(\mathcal{T}_S) = \max_{e \in E(\mathcal{T}_S)} \operatorname{rk}(M(\mathcal{T}_S, e))$.

Let us write $\mathscr{S}(A)$ for the set of all socket-trees of the multilinear form A. We define the *socket-width* of A to be $w(A) = \min_{\mathcal{T}_S \in \mathscr{S}(A)} w(\mathcal{T}_S)$.

The rest of this section is devoted to proving Theorem 1.4:

▶ **Theorem 1.4.** *For every multilinear map* A, *it holds that* $c(A) \geq w(A)$.

First, we prove that without loss of generality, we may restrict our attention to forms.

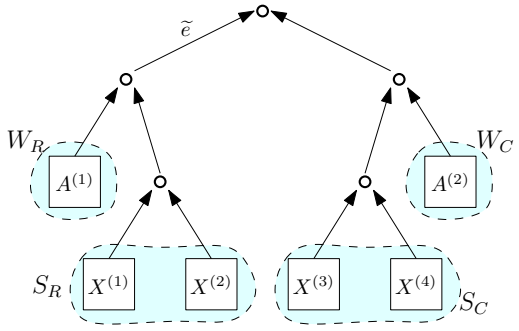
▶ **Claim 5.1.** *For any multilinear map* A, *it holds that* $c(A) \geq c(F(A))$.

Proof. We observe that A and $F(A)$ satisfy $\hat{T}(A) = \hat{T}(F(A))$. Any network $D \in \mathscr{D}(A)$ can be modified to a network $D' \in \mathscr{D}(F(A))$ by attaching a tensor $X' \in \mathbb{F}^{J(E')}$ to the boundary of D. Let $D \in \mathscr{D}(A)$ be such that $c(D) = c(A)$. The minimum-cost execution of D, followed by contracting $T(D)$ and X', is an execution of D'. Its cost is $c(A)$, since the cost of contracting of $T(D)$ and X' is $\prod_{e \in B(D)} |J(e)|$ and $\prod_{e \in B(D)} |J(e)| \leq c(A)$, because the last step of the minimum-cost execution of D contracted a set W with all modes $e \in B(D)$ incident to W. Thus, $c(A) \geq c(F(A))$. ◀

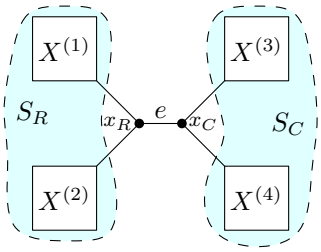
Furthermore, $w(A) = w(F(A))$ for every multilinear map A, since $w(A)$ only depends on the tensor $T(A)$, but not on which of its coordinates (if any) is the output. Thus it suffices to prove Theorem 1.4 for multilinear forms, which we now proceed to do.

▶ **Lemma 5.2.** *For any multilinear form* F, *it holds that* $c(F) \geq w(F)$.

Proof. Let $D \in \mathscr{D}(F)$ be such that $c(D) = c(F)$. It is a tensor network with empty boundary and a socket vertex $S_i \in V(D)$ for each input socket E_i, where $i = 1, 2, \ldots, \ell$. Its tensor is $T(D) = F(X^{(1)}, X^{(2)}, \ldots, X^{(\ell)})$ where $X^{(i)} = T(S_i)$ for $i = 1, 2, \ldots, \ell$.



(a) Example of a possible execution tree $\mathcal{T}_D$. Given the choice of e in the corresponding socket tree $\mathcal{T}_S$ shown on the right there are four possible choices of $\widetilde{e}$.



(b) The corresponding socket tree $\mathcal{T}_S$. The exact choice of $\widetilde{e}$ in $\mathcal{T}_D$ determines which part of the cut is the x_R part, and which is the x_C part.

Figure 1 Illustration of the notation used for the execution and socket trees.

By Lemma 3.1, a minimum-cost execution of D can be represented by a rooted binary tree $\mathcal{T}_D$, where the set of leaves of $\mathcal{T}_D$ are $V(D)$ and each inner vertex represents the vertex obtained by contracting its two children. Let $\mathcal{T}_S$ be the unique socket-tree of F that is obtained as a topological minor of $\mathcal{T}_D$. Slightly abusing the notation, we assume that the leaves of $\mathcal{T}_S$ are the socket vertices $S_1, S_2, \ldots, S_\ell$ instead of the sockets $E_1, E_2, \ldots, E_\ell$. To establish the lemma, it suffices to show that $\mathcal{T}_D$ has cost at least $w(\mathcal{T}_S)$, since $w(\mathcal{T}_S) \geq w(F)$.

Let $e = \{x_R, x_C\} \in E(\mathcal{T}_S)$ be an edge of the socket tree $\mathcal{T}_S$ with $\mathrm{rk}(M(\mathcal{T}_S, e)) = w(\mathcal{T}_S)$, and let $\widetilde{e}$ be an edge of the execution tree $\mathcal{T}_D$ in the subdivision of e appearing in $\mathcal{T}_D$. Without loss of generality we may assume that $\widetilde{e}$ is directed from the part of $\mathcal{T}_D$ corresponding to x_R towards the part corresponding to x_C (if not, simply switch names of x_R and x_C). Define $S_R = L(x_R, e)$ and $S_C = L(x_C, e)$. Let $W_R \subseteq V(D)$ be the set of non-socket vertices of D that appear on the same side of $\widetilde{e}$ in $\mathcal{T}_D$ with socket vertices S_R and let W_C be the set of remaining non-socket vertices of D. See Figure 1 for an illustration of all these definitions. Finally, let $D' = D/S_R/S_C/W_R/W_C$ be the result of contracting each of these four sets of vertices of D. For notational convenience, we identify the four vertices of the new network with the four subsets S_R, S_C, W_R, W_C.

Now, the tensor $P = T(D'[W_R \cup S_R])$ appears as an intermediate result in the execution $\mathcal{T}_D$,[5] hence the volume of P is a lower bound on the cost of $\mathcal{T}_D$.

We group the modes of D' incident on S_R or W_R as shown in Figure 2: E_{SW} are all modes in D' incident exactly upon S_R and W_R, E_{WC} are all modes incident on W_R but not on S_R, E_{SC} are all modes incident on S_R but not W_R, and finally E_{SWC} are all modes incident upon S_R, W_R, and at least one of S_C or W_C. Write $E_S = E_{SW} \cup E_{SC} \cup E_{SWC}$ for the modes incident on S_R, and similarly $E_C = E_{WC} \cup E_{SC} \cup E_{SWC}$ for all modes incident upon at least one of S_R/W_R and at least one of S_C/W_C. Note that $|J(E_C)|$ is precisely the volume of P which we aim to lower bound.

Define a matrix $A \in \mathbb{F}^{J(E_S)} \times \mathbb{F}^{J(E_C)}$ as follows. We identify its row indices $i \in J(E_S)$ as being triples $i = (i_{SW}, i_{SC}, i_{SWC}) \in J(E_{SW}) \times J(E_{SC}) \times J(E_{SWC})$ and similarly its column indices $j \in J(E_C)$ are triples $j = (j_{SC}, j_{WC}, j_{SWC}) \in J(E_{SC}) \times J(E_{WC}) \times J(E_{SWC})$. Then

[5] Note that the same is not true for the tensor $T(D'[W_C \cup S_C])$.

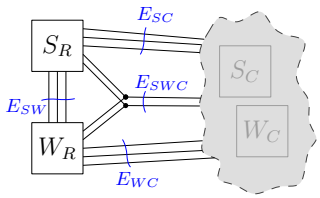


Figure 2 Illustration of D'. We group the modes of D' based on how they connect S_R, S_C, and the "C part" of D'.

the entries of A are

$$A_{(i_{SW},i_{SC},i_{SWC}),(j_{SC},j_{WC},j_{SWC})} = \begin{cases} T(D'[W_R])_{i_{SW},j_{WC},j_{SWC}} & \text{if } i_{SC} = j_{SC} \wedge i_{SWC} = j_{SWC}, \\ 0 & \text{otherwise,} \end{cases}$$

In the case when $E_S = E_{SW}$ (i.e., all modes incident on S_R connect only to W_R), A is simply a flattening of $T(D'[W_R])$. Recall that $T(D'[S_R]) \in \prod_{e \in E_S} \mathbb{F}^{J(e)}$. Then for every $j = (j_{SC}, j_{WC}, j_{SWC}) \in J(E_C)$, we have

$$\begin{aligned} \sum_{i \in J(E_S)} A_{i,j} T(D'[S_R])_i &= \sum_{i_{SW} \in J(E_{SW})} A_{(i_{SW},j_{SC},j_{SWC}),j} T(D'[S_R])_{i_{SW},j_{SC},j_{SWC}} \\ &= \sum_{i_{SW}} T(D'[W_R])_{i_{SW},j_{WC},j_{SWC}} T(D'[S_R])_{i_{SW},j_{SC},j_{SWC}} \\ &= P_{j_{SC},j_{WC},j_{SWC}} = P_j \end{aligned}$$

(recall that P is the contraction of $T(D'[W_R])$ and $T(D'[S_R])$). Viewing $T(D'[S_R])$ as a row vector in $\mathbb{F}^{J(E_S)}$ we see that P is the vector-matrix product $P = T(D'[S_R]) \cdot A \in \mathbb{F}^{J(E_C)}$.

Symmetrically, for the other half of D', we can write $Q = T(D'[W_C \cup S_C])$ as a matrix-vector product $Q = B \cdot T(D'[S_C]) \in \mathbb{F}^{J(E_C)}$ where B is a matrix corresponding to $T(D'[W_S])$ analogously to how A corresponds to $T(D'[W_R])$.

Thus we have $T(D) = T(D'[S_R]) \cdot A \cdot B \cdot T(D'[S_C])$. Recall that for each socket vertex S_i in the original network D, we have $T(S_i) = X^{(i)}$. Denoting $X_R = T(D'[S_R])$ and $X_C = T(D'[S_C])$, we get $X_R = \bigotimes_{S_i \in S_R} X^{(i)}$ and $X_C = \bigotimes_{S_i \in S_C} X^{(i)}$.[6] Hence

$$F(X^{(1)}, X^{(2)}, \ldots, X^{(\ell)}) = X_R \cdot A \cdot B \cdot X_C.$$

It follows that $A \cdot B$ is the flattening of $T(F)$ to a matrix with rows indexed by the sockets in S_R and columns indexed by the sockets in S_C. But this flattening is precisely the matrix $M(\mathcal{T}_S, e)$, implying that $|J(E_C)| \geq \mathrm{rk}(M(\mathcal{T}_S, e)) = w(\mathcal{T}_S)$, as desired. ◀

References

1 Amir Abboud, Arturs Backurs, Karl Bringmann, and Marvin Künnemann. Fine-Grained Complexity of Analyzing Compressed Data: Quantifying Improvements over Decompress-and-Solve. In *Proc. of Foundations of Computer Science (FOCS)*, pages 192–203, 2017. doi:10.1109/FOCS.2017.26.

[6] These identities use the fact that D is derived from a *non-degenerate* network D^* for $\hat{T}(F)$. In particular, every mode in the network D is incident upon at least one non-socket vertex, hence all modes incident upon S_R are boundary modes in $D'[S_R]$, and similarly for S_C.

2 Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. If the Current Clique Algorithms are Optimal, So is Valiant's Parser. In *Proc. of Foundations of Computer Science (FOCS)*, pages 98–117, 2015. doi:10.1109/FOCS.2015.16.

3 Michael Alekhnovich, Allan Borodin, Joshua Buresh-Oppenheim, Russell Impagliazzo, Avner Magen, and Toniann Pitassi. Toward a Model for Backtracking and Dynamic Programming. *Comput. Complex.*, 20(4):679–740, 2011. doi:10.1007/s00037-011-0028-y.

4 Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010. doi:10.1145/1798596.1798607.

5 Noga Alon, Raphael Yuster, and Uri Zwick. Finding and Counting Given Length Cycles. *Algorithmica*, 17(3):209–223, 1997. doi:10.1007/BF02523189.

6 Itai Arad and Zeph Landau. Quantum computation and the evaluation of tensor networks. *SIAM J. Comput.*, 39(7):3089–3121, 2010. doi:10.1137/080739379.

7 Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009. doi:10.1017/CBO9780511804090.

8 Martin Beaudry and Markus Holzer. The Complexity of Tensor Circuit Evaluation. *Comput. Complex.*, 16(1):60–111, 2007. doi:10.1007/s00037-007-0222-0.

9 Stuart J. Berkowitz. On Computing the Determinant in Small Parallel Time Using a Small Number of Processors. *Inf. Process. Lett.*, 18(3):147–150, 1984. doi:10.1016/0020-0190(84)90018-8.

10 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proc. of Symposium on Theory of Computing (STOC)*, pages 67–74, 2007. doi:10.1145/1250790.1250801.

11 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Computing the Tutte Polynomial in Vertex-Exponential Time. In *Proc. of Foundations of Computer Science (FOCS)*, pages 677–686, 2008. doi:10.1109/FOCS.2008.40.

12 Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Counting Paths and Packings in Halves. In *Proc. of European Symposium on Algorithms (ESA)*, pages 578–586, 2009. doi:10.1007/978-3-642-04128-0_52.

13 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009. doi:10.1137/070683933.

14 Andreas Björklund, Petteri Kaski, and Łukasz Kowalik. Counting Thin Subgraphs via Packings Faster Than Meet-in-the-Middle Time. In *Proc. of Symposium on Discrete Algorithms (SODA)*, pages 594–603, 2014. doi:10.1137/1.9781611973402.45.

15 Hans L. Bodlaender, Erik Jan van Leeuwen, Johan M. M. van Rooij, and Martin Vatshelle. Faster Algorithms on Branch and Clique Decompositions. In *Proc. of Mathematical Foundations of Computer Science (MFCS)*, pages 174–185, 2010. doi:10.1007/978-3-642-15155-2_17.

16 James R. Bunch and John E. Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Math. Comp.*, 28:231–236, 1974. doi:10.2307/2005828.

17 Jin-Yi Cai, Heng Guo, and Tyson Williams. A Complete Dichotomy Rises from the Capture of Vanishing Signatures. *SIAM J. Comput.*, 45(5):1671–1728, 2016. doi:10.1137/15M1049798.

18 Jin-yi Cai, Pinyan Lu, and Mingji Xia. Computational Complexity of Holant Problems. *SIAM J. Comput.*, 40(4):1101–1132, 2011. doi:10.1137/100814585.

19 Florent Capelli, Arnaud Durand, and Stefan Mengel. The Arithmetic Complexity of Tensor Contraction. *Theory Comput. Syst.*, 58(4):506–527, 2016. doi:10.1007/s00224-015-9630-8.

20 A. Cayley. On the theory of the analytical forms called trees. *Philos. Mag.*, 13:172–176, 1857. doi:10.1080/14786445708642275.

21 A. Cayley. On the analytical forms called trees, part II. *Philos. Mag.*, 18:374–378, 1859. doi:10.1080/14786445908642782.

22 A. Clebsch. Ueber symbolische Darstellung algebraischer Formen. *J. Reine Angew. Math.*, 59:1–62, 1861. doi:10.1515/crll.1861.59.1.

23 W. Clifford. Extract of a Letter to Mr. Sylvester from Prof. Clifford of University College, London. *Amer. J. Math.*, 1(2):126–128, 1878. doi:10.2307/2369303.

24 James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965. doi:10.2307/2003354.

25 Radu Curticapean, Holger Dell, and Dániel Marx. Homomorphisms are a good basis for counting small subgraphs. In *Proc. of Symposium on Theory of Computing (STOC)*, pages 210–223, 2017. doi:10.1145/3055399.3055502.

26 Carsten Damm, Markus Holzer, and Pierre McKenzie. The complexity of tensor calculus. *Comput. Complex.*, 11(1-2):54–89, 2002. doi:10.1007/s00037-000-0170-4.

27 Sashka Davis and Russell Impagliazzo. Models of Greedy Algorithms for Graph Problems. *Algorithmica*, 54(3):269–317, 2009. doi:10.1007/s00453-007-9124-4.

28 Josep Díaz, Maria J. Serna, and Dimitrios M. Thilikos. Counting H-colorings of partial k-trees. *Theor. Comput. Sci.*, 281(1-2):291–309, 2002. doi:10.1016/S0304-3975(02)00017-8.

29 Frederic Dorn. Dynamic Programming and Fast Matrix Multiplication. In *Proc. of European Symposium on Algorithms (ESA)*, pages 280–291, 2006. doi:10.1007/11841036_27.

30 Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoret. Comput. Sci.*, 326(1-3):57–67, 2004. doi:10.1016/j.tcs.2004.05.009.

31 Norman P. Jouppi et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proc. of International Symposium on Computer Architecture (ISCA)*, pages 1–12, 2017. doi:10.1145/3079856.3080246.

32 Peter Floderus, Miroslaw Kowaluk, Andrzej Lingas, and Eva-Marta Lundell. Detecting and Counting Small Pattern Graphs. *SIAM J. Discrete Math.*, 29(3):1322–1339, 2015. doi:10.1137/140978211.

33 Fedor V. Fomin, Daniel Lokshtanov, Venkatesh Raman, Saket Saurabh, and B. V. Raghavendra Rao. Faster algorithms for finding and counting subgraphs. *J. Comput. Syst. Sci.*, 78(3):698–706, 2012. doi:10.1016/j.jcss.2011.10.001.

34 William I. Gasarch. The Second P =? NP Poll. *SIGACT News Complexity Theory Column*, 74, 2012. doi:10.1145/2261417.2261434.

35 Russell Impagliazzo and Ramamohan Paturi. On the Complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001. doi:10.1006/jcss.2000.1727.

36 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001. doi:10.1006/jcss.2001.1774.

37 Alon Itai and Michael Rodeh. Finding a Minimum Circuit in a Graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.

38 Matti Karppa, Petteri Kaski, and Jukka Kohonen. A Faster Subquadratic Algorithm for Finding Outlier Correlations. *ACM Trans. Algorithms*, 14(3):31:1–31:26, 2018. doi:10.1145/3174804.

39 L. H. Kauffman. Knots, abstract tensors and the Yang-Baxter equation. In *Knots, topology and quantum field theories*, pages 179–334. World Scientific, 1989.

40 A. B. Kempe. On the Application of Clifford's Graphs to Ordinary Binary Quantics. *P. Lond. Math. Soc.*, 17:107–121, 1885/86. doi:10.1112/plms/s1-17.1.107.

41 A. B. Kempe. On the application of the Sylvester-Clifford Graphs to Ordinary Binary Quantics. (Second Part.). *P. Lond. Math. Soc.*, 24:97–118, 1892/93. doi:10.1112/plms/s1-24.1.97.

42 Ton Kloks, Dieter Kratsch, and Haiko Müller. Finding and counting small induced subgraphs efficiently. *Inf. Process. Lett.*, 74(3-4):115–121, 2000. doi:10.1016/S0020-0190(00)00047-8.

43 Tamara G. Kolda. Multilinear operators for higher-order decompositions. Technical Report SAND2006-2081, Sandia National Laboratories, 2006. doi:10.2172/923081.

44 Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009. doi:10.1137/07070111X.

45 Daphne Koller and Nir Friedman. *Probabilistic Graphical Models*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, 2009.

46 Joseph B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra Appl.*, 18(2):95–138, 1977. doi:10.1016/0024-3795(77)90069-6.

47 Greg Kuperberg. Involutory Hopf algebras and 3-manifold invariants. *Internat. J. Math.*, 2(1):41–66, 1991. doi:10.1142/S0129167X91000053.

48 J. M. Landsberg. *Tensors: Geometry and Applications*, volume 128 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2012. doi:10.1090/gsm/128.

49 Joseph M. Landsberg, Yang Qi, and Ke Ye. On the geometry of tensor network states. *Quantum Inf. Comput.*, 12(3-4):346–354, 2012.

50 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proc. of International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

51 James R. Lee, Prasad Raghavendra, and David Steurer. Lower Bounds on the Size of Semidefinite Programming Relaxations. In *Proc. of Symposium on Theory of Computing (STOC)*, pages 567–576, 2015. doi:10.1145/2746539.2746599.

52 Andrea Lincoln, Virginia Vassilevska Williams, and R. Ryan Williams. Tight Hardness for Shortest Cycles and Paths in Sparse Graphs. In *Proc. of Symposium on Discrete Algorithms (SODA)*, pages 1236–1252, 2018. doi:10.1137/1.9781611975031.80.

53 Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S. Vetter. NVIDIA tensor core programmability, performance & precision. In *Proc. of International Parallel and Distributed Processing Symposium Workshops (IPDPS)*, pages 522–531, 2018. doi:10.1109/IPDPSW.2018.00091.

54 J. Nešetřil and S. Poljak. On the complexity of the subgraph problem. *Comment. Math. Univ. Carolin.*, 26(2):415–419, 1985.

55 Stephan Olariu. Paw-Free Graphs. *Inf. Process. Lett.*, 28(1):53–54, 1988. doi:10.1016/0020-0190(88)90143-3.

56 R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Ann. Phys.*, 349:117–158, 2014. doi:10.1016/j.aop.2014.06.013.

57 Roger Penrose. *Tensor Methods in Algebraic Geometry*. PhD thesis, St. John's College, 1956.

58 Roger Penrose. Applications of negative dimensional tensors. In *Combinatorial Mathematics and its Applications*, pages 221–244. Academic Press, 1971.

59 Robert N. C. Pfeifer, Jutho Haegeman, and Frank Verstraete. Faster identification of optimal contraction sequences for tensor networks. *Phys. Rev. E*, 90:033315, 2014. doi:10.1103/PhysRevE.90.033315.

60 Ran Raz. Tensor-Rank and Lower Bounds for Arithmetic Formulas. *J. ACM*, 60(6):40:1–40:15, 2013. doi:10.1145/2535928.

61 Jürgen Richter-Gebert and Peter Lebmeir. Diagrams, tensors and geometric reasoning. *Discrete Comput. Geom.*, 42(2):305–334, 2009. doi:10.1007/s00454-009-9188-9.

62 Elina Robeva and Anna Seigal. Duality of graphical models and tensor networks. *Inf. Inference*, page iay009, 2018. doi:10.1093/imaiai/iay009.

63 Günter Rote. Division-Free Algorithms for the Determinant and the Pfaffian: Algebraic and Combinatorial Approaches. In *Computational Discrete Mathematics, Advanced Lectures*, pages 119–135, 2001. doi:10.1007/3-540-45506-X_9.

64 Herbert John Ryser. *Combinatorial Mathematics*. The Carus Mathematical Monographs, No. 14. Mathematical Association of America, 1963.

65 Alexander Schrijver. On traces of tensor representations of diagrams. *Linear Algebra Appl.*, 476:28–41, 2015. doi:10.1016/j.laa.2015.02.037.

66 Edgar Solomonik and Torsten Hoefler. Sparse Tensor Algebra as a Parallel Programming Model. *arXiv:1512.00066*, 2015.

67 Edgar Solomonik, Devin Matthews, Jeff R. Hammond, John F. Stanton, and James Demmel. A massively parallel tensor contraction framework for coupled-cluster computations. *J. Parallel Distrib. Comput.*, 74(12):3176–3190, 2014. doi:10.1016/j.jpdc.2014.06.002.

68 Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969. doi:10.1007/BF02165411.

69 J. J. Sylvester. On an Application of the New Atomic Theory to the Graphical Representation of the Invariants and Covariants of Binary Quantics, with Three Appendices. *Amer. J. Math.*, 1(1):64–125, 1878. doi:10.2307/2369436.

70 Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979. doi:10.1016/0304-3975(79)90044-6.

71 Leslie G. Valiant. Holographic Algorithms. *SIAM J. Comput.*, 37(5):1565–1594, 2008. doi:10.1137/070682575.

72 Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM, 1992. doi:10.1137/1.9781611970999.

73 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proc. of Symposium on Theory of Computing (STOC)*, pages 887–898, 2012. doi:10.1145/2213977.2214056.

74 Virginia Vassilevska Williams and Ryan Williams. Finding, Minimizing, and Counting Weighted Subgraphs. *SIAM J. Comput.*, 42(3):831–854, 2013. doi:10.1137/09076619X.

75 Ryan Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.*, 348(2-3):357–365, 2005. doi:10.1016/j.tcs.2005.09.023.

76 Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proc. of Symposium on Theory of Computing (STOC)*, pages 664–673, 2014. doi:10.1145/2591796.2591811.

77 Virginia Vassilevska Williams, Joshua R. Wang, Richard Ryan Williams, and Huacheng Yu. Finding Four-Node Subgraphs in Triangle Time. In *Proc. of Symposium on Discrete Algorithms (SODA)*, pages 1671–1680, 2015. doi:10.1137/1.9781611973730.111.